Population-Based Black-Box Optimization for Biological Sequence Design

Christof Angermueller [1] **David Belanger** [1] **Andreea Gane** [1] **Zelda Mariet** [1] **David Dohan** [1] **Kevin Murphy** [1] **Lucy Colwell** [1 2] **D Sculley** [1]

Abstract

The use of black-box optimization for the design of new biological sequences is an emerging research area with potentially revolutionary impact. The cost and latency of wet-lab experiments requires methods that find good sequences in few experimental rounds of large batches of sequences — a setting that off-the-shelf black-box optimization methods are ill-equipped to handle. We find that the performance of existing methods varies drastically across optimization tasks, posing a significant obstacle to real-world applications. To improve robustness, we propose Population-Based Black-Box Optimization (P3BO), which generates batches of sequences by sampling from an ensemble of methods. The number of sequences sampled from any method is proportional to the quality of sequences it previously proposed, allowing P3BO to combine the strengths of individual methods while hedging against their innate brittleness. Adapting the hyper-parameters of each of the methods online using evolutionary optimization further improves performance. Through extensive experiments on *in-silico* optimization tasks, we show that P3BO outperforms any single method in its population, proposing higher quality sequences as well as more diverse batches. As such, P3BO and Adaptive-P3BO are a crucial step towards deploying ML to real-world sequence design.

1. Introduction

The ability to design new protein or DNA sequences with desired properties would revolutionize drug discovery, healthcare, and agriculture. However, this is a particularly challenging optimization problem: the space of sequences is discrete and exponentially large; evaluating the fitness of proposed sequences requires costly wet-lab experiments; and not just one but a *diverse* set of high-quality sequences must be discovered to improve the chance of a candidate surviving downstream screening (*e.g.*, for toxicity).

The problem is not insurmountable, however, due to modern experimental technology, where hundreds to thousands of sequences can be evaluated in parallel. This forms the basis for directed evolution, a form of human-guided local evolutionary search (Arnold, 1998), and several ML-based methods (Section 4). Additionally, development of ML methods suitable to guide sequence design can be aided by working *in-silico*, instead of relying on wet-lab processes during algorithmic development.

In this paper, we introduce several *in-silico* design problems upon which we evaluate such ML methods. Unfortunately, we find that popular optimization methods are particularly sensitive to hyper-parameter choice and can have strong inductive biases that allow them to excel at some problems but perform poorly on others (Figure 1). This lack of robustness is a serious concern for practical application of these methods, which could cause wet-lab experiments to fail. We further find that several existing methods are ill-suited to generating diverse batches of sequences. Instead of using the batch size efficiently, methods tend to generate very similar sequences.

To improve robustness and sequence diversity, we introduce Population-Based Black-box Optimization (P3BO). P3BO draws inspiration from portfolio algorithms (Leyton-Brown et al., 2003; Tang et al., 2014) for numerical optimization: instead of generating a batch of sequences using a single potentially brittle algorithm, P3BO samples sequences from a *portfolio* of algorithms, allocating budget to each algorithm based on the quality of its past proposed sequences.

To our knowledge, P3BO is the first approach to leverage an ensemble of optimizers for *batched* optimization, a crucial characteristic of wet-lab optimization loops. We show that batching offers a dimension along which ensembling yields significant gains.

Notably, samples acquired by one algorithm are shared with all other algorithms, allowing each of them to learn from data acquired by all. By combining the strengths of multiple

[1]Google Research [2]University of Cambridge. Correspondence to: Christof Angermueller <christofa@google.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

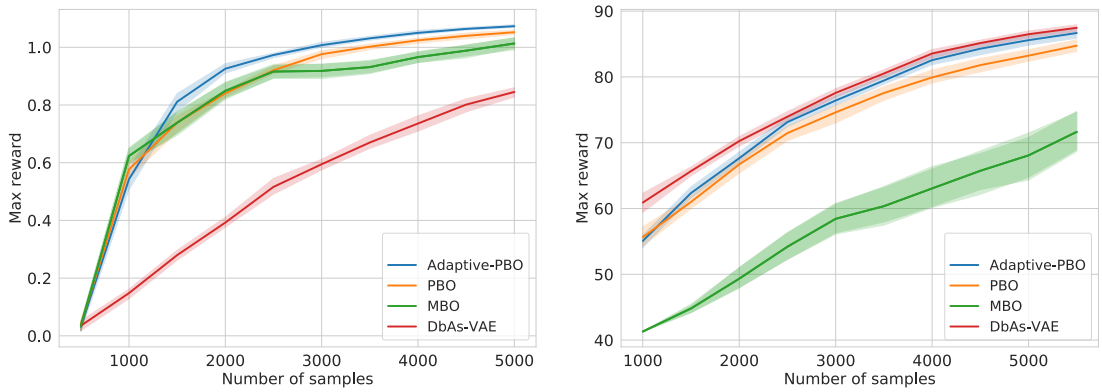


Figure 1: Motivating example: comparison of optimization trajectories for baseline ML methods and P3BO on optimization problems (Section 5) with qualitatively-different functional forms: PdbIsing (left) and PfamHMM (right). The rank ordering of performance among the baseline methods for these two seemingly similar problems is inverted, showing the potential brittleness of existing approaches for biological sequence design.

algorithms, P3BO hedges against the risk of choosing an unsuitable optimization algorithm for the problem at hand (Section 2.2). Sampling sequences from multiple algorithms additionally allows P3BO to produce more diverse batches and find distinct optima faster. We employ a heterogenous population, consisting of global model-based optimizers based on discriminative and generative models along evolutionary strategies. Finally, we further improve P3BO by introducing a variant, Adaptive-P3BO, which adapts the hyper-parameters of the algorithms themselves on the fly using evolutionary search.

We evaluate P3BO and Adaptive-P3BO empirically on over 100 batched black-box optimization problems, and show that P3BO and Adaptive-P3BO are considerably more robust, generate more diverse batches of sequences, and find distinct optima faster than any single method in their population. Adaptive-P3BO improves upon P3BO results, and furthermore is able to recover from a poor initial population of methods. Our contributions are as follows:

- We introduce new in-silico optimization problems for benchmarking biological sequence design methods.
- We evaluate state-of-the-art sequence design methods across these problems, bringing to light two significant shortcomings of existing methods: (a) lack of generalization across similar problem classes, and (b) sub-optimal use of the large batch sizes crucial to wet-lab settings.
- We introduce P3BO: a population-based optimization framework for discrete batched black-box function optimization that ensembles over algorithms to hedge against brittleness and improve diverse sequence discovery.
- We introduce Adaptive-P3BO, an extension of P3BO that tunes the hyper-parameters of population members using evolutionary search, yielding further improvements upon P3BO.

2. Problem Setting and Motivation

We define sequences $\mathbf{x}$ as elements of $\mathcal{V}^L$, where $\mathcal{V}$ is a finite vocabulary (for DNA, $|\mathcal{V}| = 4$; for proteins, $|\mathcal{V}| = 20$) and L is the sequence length. For variable length sequences, we assume that sequences are padded to length L by an end-of-sequence token.

Sequence design aims to maximize a function $f : \mathcal{V}^L \to \mathbb{R}$, which can be evaluated on batches of sequences $\mathcal{X} \subseteq \mathcal{V}^L$ size $B = |\mathcal{X}|$, but only a limited number of times T.

2.1. Algorithm Requirements

Most discrete black-box optimization methods have associated hyper-parameters. Throughout the paper, an *algorithm* A refers to an instance of a particular method class (*e.g.*, evolutionary search), which is instantiated by a specific hyper-parameter configuration (*e.g.*, mutation rate for evolutionary search). As P3BO ensembles heterogeneous algorithms, including global model-based optimizers and local search strategies, we make the following assumptions about the interface of algorithms.

$A.\text{fit}(\mathcal{X}, \mathcal{Y})$ updates its internal state (*e.g.*, ML model) using a batch of sequences $\mathcal{X}$ and their objective values $\mathcal{Y} = \{f(\mathbf{x}) \mid \mathbf{x} \in \mathcal{X}\}$. Next, $A.\text{propose}()$ suggests a single sequence to be evaluated next. We require that $\text{fit}(\mathcal{X}, \mathcal{Y})$ can leverage data obtained by other algorithms, which prohibits the use of on-policy RL methods.

2.2. Robustness of Black-Box Optimizers

Figure 1 demonstrates the lack of robustness of existing optimization methods (Section 6.1) on two representative in-silico optimization problems (Section 5). On PDBIsing (left), model-based optimization (MBO), an optimizer based on a discriminative model of $f(\mathbf{x})$, performs well, and DbAs-VAE, which employs a generative model, struggles. On PfamHMM, their relative performance reverses order.

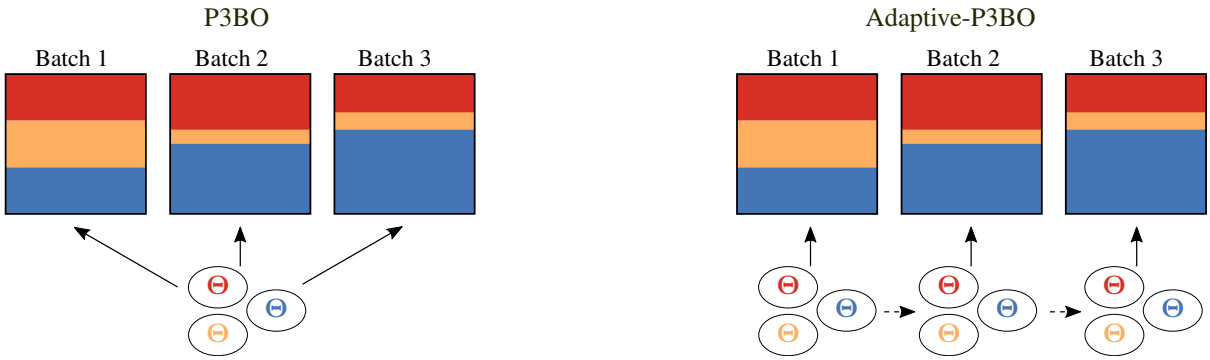


Figure 2: P3BO (left) adjusts the fraction of each batch that is allocated to a particular algorithm in its population based on the quality of sequences that the algorithm proposed in the past. Adaptive-P3BO (right) additionally adapts the hyper-parameters Θ of algorithms by evolutionary search.

This is in large part due to differences in the compatibility of methods' inductive biases with the form of $f(\mathbf{x})$. On PDBIsing, the objective function is the sum of local terms at each sequence position and long-range pairwise interactions between positions. Given limited samples, the discriminative model can estimate both of these terms, and propose high-quality sequences accordingly. On PfamHMM, the objective function is the likelihood of a generative model fit to data with insertions and deletions. A optimization method based on a generative model can capture such variability. In real applications, it is crucial that methods are robust to the structure of $f(\mathbf{x})$. P3BO, our proposed population-based approach, performs at least as well as the best algorithm in its population and often exceeds its performance.

3. Population-Based Optimization for Batched Sequence Design

We introduce P3BO, a robust black-box optimization method that constructs batches of sequences with which to query $f(\mathbf{x})$ using a population of heterogeneous optimization algorithms. By sharing data between algorithms, algorithms benefit from each other's distinct exploration strategies.

3.1. Method Summary

The high-level structure of P3BO is summarized in Algorithm 1 and Figure 2. The input is an initial population $\mathcal{P}^0 = \{A_1, ..., A_N\}$ of N constituent algorithms, which can be sampled from a distribution over algorithms or chosen using prior knowledge. At experimental round t, P3BO constructs a batch $\mathcal{X}^t$ of B sequences by iteratively sampling algorithm an A_i from a categorical distribution parameterized by p^t. The sampled algorithm A_i is then used to propose a sequences $\mathbf{x}$, which is added to the batch $\mathcal{X}^t$. Note that the constructed batch is a set $\mathcal{X}^t$ of *unique* sequences, assuring that $f(\mathbf{x})$ is not evaluated on identical sequences, which would be a waste of resources.

P3BO weights algorithms proportional to the quality of sequences that they found in the past, sampling more sequences from algorithms that found higher quality sequences. This is done by computing a reward r_i^t for each algorithm A_i at each step t and adjusting the probability of sampling from A_i based on r_i^t. To evaluate the performance of each algorithm in the population, P3BO keeps track of which algorithms proposed each sequence in $\mathcal{X}^t$ using subsets $\mathcal{X}_i^t$. If two algorithms A_i and A_j propose the same sequence $\mathbf{x} \in \mathcal{X}^t$, $\mathbf{x}$ is added to both $\mathcal{X}_i^t$ and $\mathcal{X}_j^t$.

After generating batch $\mathcal{X}^t$, the target function $f(\mathbf{x})$ is evaluated, yielding observations $\mathcal{Y}^t$. These are used to compute *rewards* for each method (Section 3.2), which are used in turn to compute sampling probabilities p^{t+1} for the next round. Finally, the fit() method of all algorithms is called on the extended data $(\mathcal{X}, \mathcal{Y})$ to update their internal state. The fitting step itself is algorithm-specific, and can entail time-consuming steps such as fitting discriminative or generative models.

3.2. Selecting from a Population of Algorithms

To generate high-quality sequences consistently across optimization rounds and different optimization problems, P3BO must adjust each constituent algorithm's contribution over time. At round t, P3BO observes all objective function values for sequences $\mathcal{X}_i^t$ that were proposed by algorithm A_i. These observations are converted into a per-algorithm reward r_i^t. Doing so is challenging because the the qualities of the sequences from each algorithm are correlated, since the algorithms share observations between optimization rounds.

In our experiments, we used the improvement of $f(\mathbf{x})$ relative to $f_{\max} = \max\{f(x) \mid x \in \mathcal{X}\}$, the maximum of $f(\mathbf{x})$ of all previous rounds:

$$r_i^t = \frac{\max\{f(\mathbf{x}) \mid \mathbf{x} \in \mathcal{X}_i^t\} - f_{\max}}{f_{\max}}. \quad (1)$$

Future work should consider alternative reward functions to (1), such as novelty search (Lehman & Stanley, 2008).

Algorithm 1 P3BO

Input: Population $\mathcal{P} = \{A_1, \ldots, A_N\}$
Input: Softmax temperature $\tau > 0$.
Input: Initial sampling weights p^1
$\mathcal{X}, \mathcal{Y} = \emptyset, \emptyset$ ▷ Sequences and labels
for $t = 1$ **to** T **do**
  $\mathcal{X}^t = \emptyset$
  $\mathcal{X}_1^t, \ldots, \mathcal{X}_N^t = \emptyset, \ldots, \emptyset$
  while $|\mathcal{X}^t| \leq B$ **do**
    $i \sim \text{Categorical}(p^t)$
    $x = A_i.\text{propose}()$
    $\mathcal{X}^t \leftarrow \mathcal{X}^t \cup \{x\}$ ▷ Only add x if novel
    $\mathcal{X}_i^t \leftarrow \mathcal{X}_i^t \cup \{x\}$ ▷ Only add x if novel
  $\mathcal{Y}^t = \{f(x) \mid x \in \mathcal{X}^t\}$
  $\mathcal{X}, \mathcal{Y} \leftarrow \mathcal{X} \cup \mathcal{X}^t, \mathcal{Y} \cup \mathcal{Y}^t$
  $r^t = \text{get\_rewards}\big(\mathcal{X}, \mathcal{Y}, \{\mathcal{X}_i^t\}_i\big)$ ▷ Eq. 1
  $s^t = \text{decayed\_rewards}(r^t)$ ▷ Eq. 2
  if Adaptive P3BO **then**
    $\mathcal{P}, s^t = \text{adapt}(\mathcal{P}^t, s^t)$ ▷ Alg. 2
  $p^{t+1} = \text{softmax}(\hat{s}^t/\tau)$ ▷ Eq. 3
  for $A_i \in \mathcal{P}$ **do**
    $A_i.\text{fit}(\mathcal{X}, \mathcal{Y})$
return X

As P3BO aims to address lack of robustness in existing methods, we prefer algorithms that consistently propose good sequences. To do so, the probability p_i of sampling algorithm A_i depends not only on its reward at time t but the sum of exponentially decayed rewards via a *credit score* s_i^t:

$$s_i^t = \sum_{t \leq t} r_i^t \gamma^{t-t}, \tag{2}$$

$$p_i = \frac{\exp(\hat{s}_i/\tau)}{\sum_j \exp(\hat{s}_j/\tau)}, \tag{3}$$

The decay rate γ trades-off past and present improvements, assigning higher credit scores to algorithms that improve $f(\mathbf{x})$ consistently across optimization rounds. $\hat{s}_i$ are min-max normalized values of the credit scores, which ensures that the p_i are independent of the scale of the rewards. The hyper-parameter τ controls the entropy of the distribution, effectively trading off the exploration and exploitation of algorithms. If τ is set to a high value, sequences are sampled uniformly from the population, regardless of their rewards.

3.3. Adaptive Population-Based Optimization

Although the credit assignment and selection strategy described in Section 3.2 increases robustness by sampling few or no sequences from poorly performing methods in the population, it is limited to the set of algorithms that are already in the population. For example, the hyper-parameters of algorithms in the population can be sub-optimal for a particular problem, which upper-bounds the performance of P3BO. We address this limitation by optimizing hyper-parameters of algorithms in the population online by evolutionary search (Algorithm 2).

We first select the set $\mathcal{S}$ of algorithms with the top-q credit scores from $\mathcal{P}$, where q is a quantile cut-off. We then use tournament selection (Miller et al., 1995) to select $k = 2$ parent algorithms from the pool of survivors $\mathcal{S}$, and recombine their hyper-parameters. If the parents belong to different classes of algorithms and their hyper-parameters are incompatible, we select one of them randomly. Otherwise, we crossover their hyper-parameters with some crossover rate. Finally, we mutate the resulting hyper-parameters with some mutation rate by either resampling hyper-parameters values from a prior distribution, or scaling them by a constant.

Algorithm 2 Adaptation of population members

Input: Population of algorithms $\mathcal{P} = \{A_1, \ldots, A_N\}$
Input: Algorithm scores $s = \{s_1, \ldots, s_N\}$
Input: Quantile cutoff q
$S = \{A_i \in \mathcal{P} \mid f_i \geq q\}$
$\widetilde{\mathcal{P}}, \widetilde{s} = \emptyset, \emptyset$
for $i = 1$ **to** N **do**
  parents = tournament_select($\mathcal{S}$)
  $A_i, s_i = \text{recombine}(\text{parents})$
  $\widetilde{A}_i = \text{mutate}(A_i)$
  $\widetilde{\mathcal{P}} \leftarrow \widetilde{\mathcal{P}} \cup \{\widetilde{A}_i\}$
  $\widetilde{s} \leftarrow \widetilde{s} \cup \{s_i\}$
return $\widetilde{\mathcal{P}}, \widetilde{s}$

4. Related Work

Our work draws from related research in both ML-guided sequence design and population-based optimization.

ML for Sequence Design. Methods based on generative models seek to maximize the expected value $\mathbb{E}_{p(\mathbf{x})}[f(\mathbf{x})]$ of the objective function $f(\mathbf{x})$ when sampling sequences $\mathbf{x}$ from a distribution $p(x)$ that is parameterized, for example, using a RNN. Most approaches for maximizing this expectation can be seen as instances of the cross-entropy method (De Boer et al., 2005; Neil et al., 2018; Brookes & Listgarten, 2018; Gupta & Zou, 2018; Brookes et al., 2019). At each round, the distribution is trained to maximize the likelihood of high-reward sequences seen so far. The next batch is sampled from this distribution.

Throughout the paper, we use 'model-based optimization' to refer to machine learning approaches that employ a discriminative surrogate model $\hat{f}(\mathbf{x})$ that approximates $f(\mathbf{x})$. The surrogate is converted into an *acquisition function*, which is optimized to propose the next batch of sequences (Hashimoto et al., 2018; Wang et al., 2019; Yang

et al., 2019; de Jongh et al., 2019; Liu et al., 2019; Sample et al., 2019; Wu et al., 2019b; Biswas et al., 2020).

Approximating $f(\mathbf{x})$ by a surrogate $\hat{f}(\mathbf{x})$ has two advantages. First, $\hat{f}(\mathbf{x})$ is inexpensive to evaluate unlike $f(x)$, which can require costly wet-lab experiments. Second, knowledge of the structure of $\hat{f}(\mathbf{x})$ or its gradients can be used to guide the optimization of the acquisition function. Existing methods differ in the form of $\hat{f}(\mathbf{x})$, the type of the acquisition function (*e.g.*, expected improvement (Mockus et al., 2014)), and the optimization of the acquisition.

Recently, Angermueller et al. (2020) proposed a hybrid approach for sequence design, where a generative policy is updated using model-based RL if the surrogate discriminative model is accurate, and model-free RL otherwise.

Population-Based Optimization. Ensembling is a common strategy to produce robust algorithms by combining diverse algorithms that have individual weaknesses. Ensembles of evolutionary and swarm algorithms have been proposed for non-batched optimization, including multi-strategy methods (Du & Li, 2008), portfolio algorithms (Leyton-Brown et al., 2003; Tang et al., 2014), and hyper-heuristics (Burke et al., 2013). Existing approaches surveyed in Wu et al. (2019a) can be categorized into low-level ensembles, which agglomerate different instances of the same class of algorithm such as different mutation operators for evolutionary search, and high-level ensembles, which operate over algorithms belonging to heterogeneous families. P3BO belongs to the later category.

Ensemble optimizers differ in how constituent algorithms are selected over time, also known as adaptive operator selection (AOS) (Maturana et al., 2009; Fialho et al., 2010; Li et al., 2013). AOS first defines the credit score of operators based on their past rewards and then selects operators based on their score. The average reward, relative reward improvement, or sum of ranks are common credit assignment strategies. Wu et al. (2019a) includes a review of such operator selection techniques, which tend to resemble the popular weighted majority method (Littlestone et al., 1989) for multi-armed bandits. AOS is not a bandit problem, however, since the action at time t impacts the rewards that different algorithms experience at later steps.

Evolutionary reinforcement learning (Pourchot & Sigaud, 2018; Khadka & Tumer, 2018) adapts a population of agents over time and exchanges observations between them. However, it is a low-level ensemble, combining homogeneous RL agents, and it performs non-batched, continuous optimization in the space of agents' parameters. Population-based training (PBT) (Jaderberg et al., 2017) jointly optimizes the weights and hyper-parameters of neural networks. However, the optimization is on a static training set, instead of data collected on-the-fly. AlphaStar applies population-based training to evolve a set of agents in a multi-agent RL setting (Vinyals et al., 2019).

5. In-Silico Benchmarking Problems

Before deploying optimization methods on expensive wet-lab experiments, it is crucial to analyze them using in-silico surrogates. This section describes the benchmark problems that we have used to study the strengths and weaknesses of different methods. See Suppl. Section A for further details.

Creating realistic problems is challenging because protein fitness landscapes have not been experimentally well-characterized, and understanding their properties is an open research question. However, we can create problems with a range of functional forms containing elements that appear in real landscapes. Our problems fall into four categories: (1) exhaustive wet-lab measurements of all sequences of a small search space, (2) regressors fit to wet-lab measurements for a subset of sequences from problems with larger search spaces, (3) neural networks with random weights, and (4) statistical models for protein sequence evolution fit using experimental data. Problems vary in the size of their search space, the number of initial samples provided to optimizers, and the sensitivity $f(\mathbf{x})$ to shifts, insertions, and deletions of $\mathbf{x}$. See Section 5 for more details about individual optimization problems.

TfBind8: Barrera et al. (2016) measured the binding activity between a variety of human transcription factors and every possible length-8 DNA sequence. For each transcription factor, the optimization goal is to identify DNA sequences that maximize the binding activity score.

TfBind10: (Le et al., 2018) provides neural network predicted estimates of the relative binding affinities between all unique length-10 DNA sequences and each of two protein targets. For each target, the goal to identify DNA sequences that maximize the predicted binding affinity.

UTR: Sample et al. (2019) introduced a CNN to predict the impact of a 5'UTR sequence on the expression level of its corresponding gene. The goal is to identify length-50 DNA sequences that maximize the predicted expression level.

RandomMLP/RandomRNN: Following Brookes & Listgarten (2018), we design optimization problems with the goal find inputs that maximize the scalar output of randomly initialized fully-connected or recurrent neural networks.

PfamHMM: Pfam (El-Gebali et al., 2018) is a widely-used database of families of protein domain sequences. Each family consists of a small set of human-curated seed sequences, along with sequences that have been added automatically based on the likelihood under a profile hidden Markov model (HMM) fit using the seed sequences (Finn

Problem	Adap. P3BO	P3BO	MBO	DbAs VAE	Latent MBO	Evo	SMW
PdbIsing	**7.0**	5.6	5.2	3.8	3.0	2.3	1.1
PfamHMM	5.8	5.0	2.4	**6.0**	3.6	2.2	3.0
ProteinDist.	**6.8**	6.2	4.8	3.0	4.1	2.1	1.0
RandomMLP	**7.0**	5.9	4.5	3.6	3.9	2.0	1.0
RandomRNN	6.2	**6.7**	5.1	1.2	2.9	3.7	2.2
TfBind10	5.0	**6.5**	**6.5**	3.0	2.0	1.0	4.0
TfBind8	**6.8**	5.9	4.5	3.0	1.3	1.9	4.6
UTR	**7.0**	6.0	2.0	4.0	1.0	5.0	3.0

Table 1: Ranking of methods in terms of the area under the *Max reward* curve per optimization class (higher is better). Shown is the mean rank over all instances per optimization problem (Suppl. Table 1). As also shown in Figure 3, both P3BO and Adaptive-P3BO outperform or are comparable to all other baseline methods for sequence optimization.

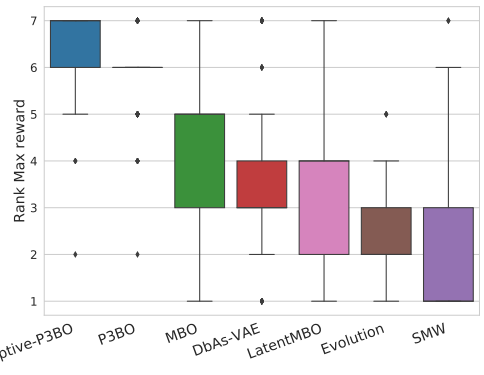


Figure 3: Ranking of methods in terms of the area under the *Max reward* curve per optimization problem (higher is better). Shown is the distribution of ranks over all instances per optimization problem (Suppl. Table 1). The box of P3BO is flat since both the 25th and 75th percentile over all 105 optimization problems is 6.

et al., 2011). We construct optimization problems for 24 diverse protein families. For each, we use the likelihood of the profile HMM as a black-box objective function. All optimization methods are provided with an initial dataset consisting of a random subset of the sequences in the family with a likelihood below the 50th percentile. This simulates practitioners' use of tools such as HMMer (Finn et al., 2011) or HHblits (Remmert et al., 2012) to find a set of evolutionarily-related sequences to inform sequence design.

ProteinDistance: Bileschi et al. (2019) introduced a CNN for protein domain classification that yields informative 1100-dimensional embeddings of protein domains. The ProteinDistance problem tasks methods to find sequences with high cosine similarity in the embedding space of this model to representative sequences from Pfam families.

PDBIsing: As introduced in Angermueller et al. (2020), the goal of this problem is to find protein sequences that maximize the energy of an Ising model parameterized by a protein structure from the Protein Data Bank (Berman et al., 2003) (PDB). We reweight the objective function to place more emphasis on long-range pairwise interactions between amino acids. Optimization methods are provided with an initial dataset of mutants of the sequence that the PDB structure is based on.

6. Experiments

We next analyze the performance of a set of competitive optimization methods on the benchmark problems described the previous section.

6.1. Baseline Optimization Methods

We consider the following methods that have been designed for batched black-box optimization of biological sequences.

SingleMutantWalker (SMW) simulates site-saturation mutagenesis, where all single-mutation neighbors of the best sequence seen so far are proposed in a given optimization round (Wu et al., 2019b).

Evolution performs directed evolutionary search by selecting the top k sequences, recombining them, and mutating them (unter Rudolph, 1958; Brindle, 1980).

Cross-Entropy Method fits a generative model to maximize the likelihood of high-quality sequences and samples the next batch of sequences from this model. **DbAs-VAE** uses the same example weighting scheme and generative model, a fully-connected variational autoencoder (Kingma & Welling, 2014), as in (Brookes & Listgarten, 2018). Supplementary material also considers **FBGAN** (Gupta & Zou, 2018), which uses a generative adversarial network.

Model-Based Optimization (MBO) automatically tunes the hyper-parameters of diverse candidate regressor models as described in (Angermueller et al., 2020). All models with cross-validation performance above a predefined threshold are ensembled, yielding a predicted mean and variance for each seqeunce. These are converted into an acquisition function (*e.g.*, expected improvement), which is optimized with regularized evolution (Real et al., 2019) to yield the next batch of sequences.

Latent-Space MBO (LatentMBO): We adapt the method of (Gómez-Bombarelli et al., 2018) to batched optimization. Model-based optimization is performed in the continuous latent space of a generative model that is trained jointly with a regressor model, which predicts $f(\mathbf{x})$ given the latent embedding of $\mathbf{x}$. We use the cross-entropy method for optimizing the regressor.

6.2. Population-Based Optimization

We used a population with $N = 15$ constituent algorithms belonging to the following classes: **SMW**, **MBO** with various regressors, acquisition functions, and mutation rates of the evolution acquisition function optimizer; **DbAs** with

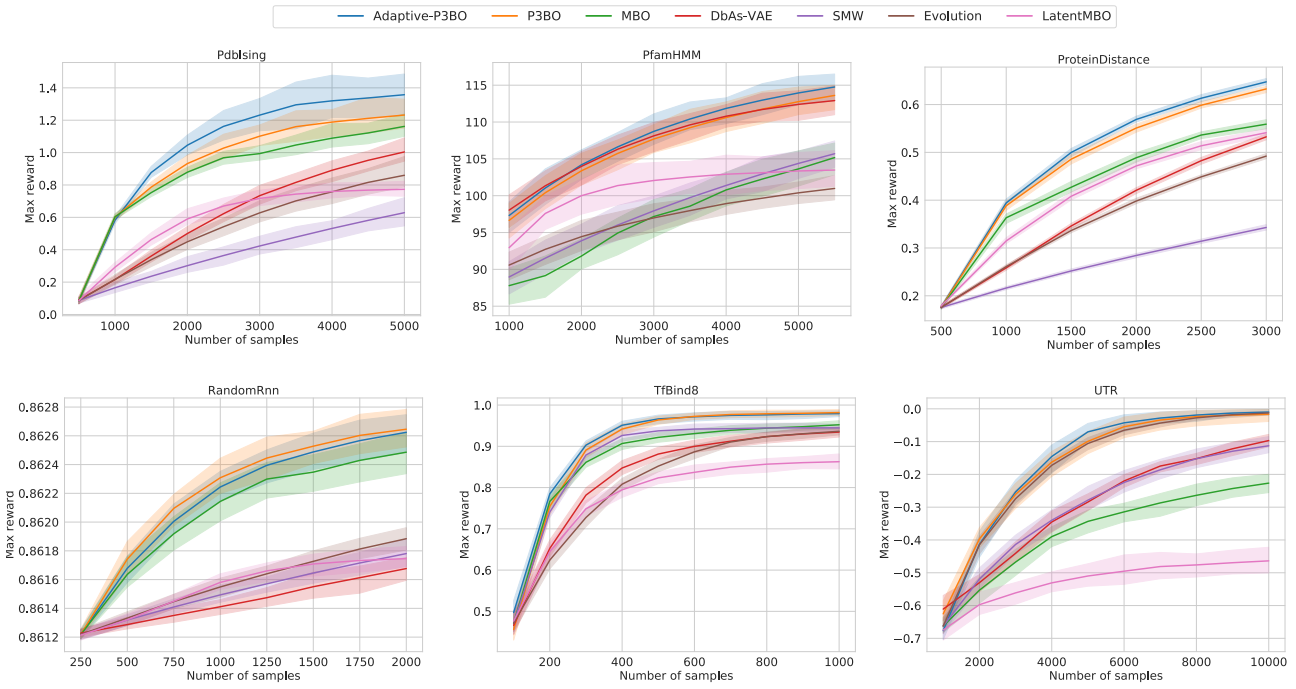


Figure 4: Performances curves for different problem classes. Lines show the average over all instances per problem class. The shaded area corresponds to the 95% bootstrap confidence interval. The synergistic effect of the ensembling by P3BO is noteworthy: not only does it match the performance of the best ensemble member, but it outperforms it. Adaptive P3BO provides a further improvements in most cases. Due to space limitations, results for RandomMLP and TfBind10 are shown in Suppl. Figure 2, and results for additional baselines in Suppl. Figure 3.

different generative models (VAE or LSTM) and quantile thresholds; and **Evolution** with varying mutation and crossover probabilities. See Suppl. Section B.7 for more details.

We sampled the initial population of algorithms randomly, ensuring that P3BO includes at least one instance per class to promote diversity, and at most four MBO instances to reduce computational costs. We used a decay factor of $\gamma = 0.25$ for computing credit scores, and a softmax temperature of $\tau = 1.0$ for computing selection probabilities.

We compare P3BO to Adaptive-P3BO, which adapts population members as described in Section 3.3. For Adaptive-P3BO, we use a quantile cutoff of $q = 0.5$ for selecting the pool of survivors $\mathcal{S}$, a recombination rate of 0.1, and a mutation rate of 0.5. Experimentally, we observed that the performance of Adaptive-P3BO remained robust to any of these hyper-parameters.

6.3. Evaluation of Sample-Efficiency and Robustness

We evaluate sample-efficiency for a particular optimization problem by comparing the cumulative maximum of $f(\mathbf{x})$ (*Max reward*) depending on the number of samples proposed. We further use the area under the *Max reward* curve to summarize sample-efficiency in a single number and comparing methods across optimization problems. We repeat each experiment 20 times with different random seeds.

Figure 4 illustrates the mean optimization trajectories of P3BO, Adaptive-P3BO, and baselines over 6 different problem classes. P3BO and Adaptive-P3BO systematically find high-reward sequences faster than any baseline method, which shows that population-based optimization can increase both sample-efficiency and robustness. Adapting the population of optimizers (Adaptive-P3BO) yields further performance improvements. Non-population-based methods have inconsistent performance: there is no best non-population-based method across problems. Optimization trajectories for additional problem classes and baseline methods are shown in Suppl. Figure 2 and Suppl. Figure 3, respectively.

Table 1 and Figure 3 summarize results, which report the average ranking of different optimization methods across all 105 optimization problems. P3BO and Adaptive-P3BO are systematically more sample-efficient than any baseline method.

6.4. Evaluation of Diversity

We evaluate the ability of methods to find distinct sequences of high reward using the metrics summarized in Suppl. Section C.

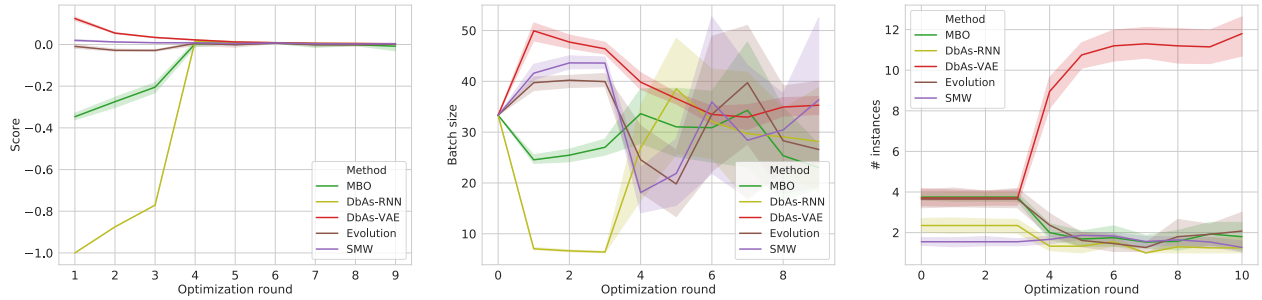
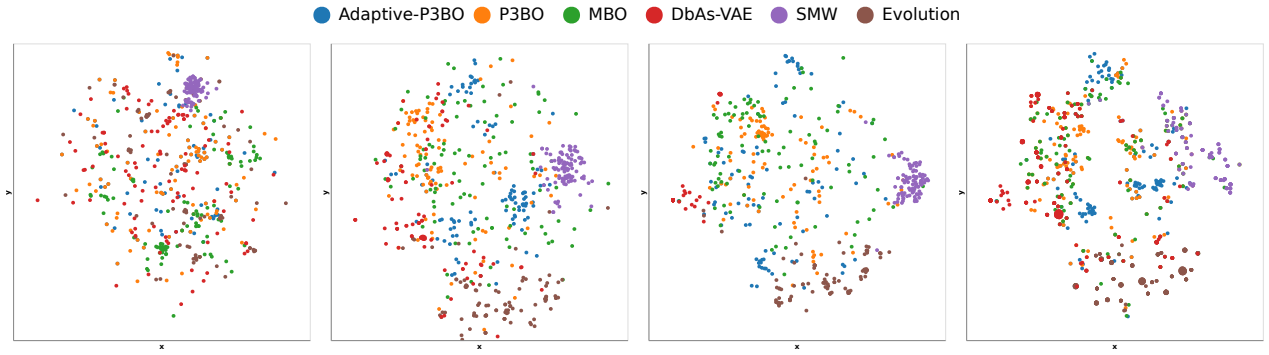


Figure 5: Insights into Adaptive-P3BO applied to PfamHMM target PF16186. Shown are the credit score (left), the number of sequences sampled (middle), and the number of instances (right) per algorithm class over time. Since DbAs-VAE has the highest credit score (relative improvement) in early rounds, more sequences are sampled from DbAs-VAE (middle), and Adaptive-P3BO increases the number of DbAs-VAE instances from 4 to 11 (the total population size is 15). The adaptation starts after three warm-up rounds used to reliably estimate the credit score of algorithms. See Suppl. Figure 4 for another example.



(a) Batch 1 (b) Batch 5 (c) Batch 9 (d) Rewards in 75th percentile

Figure 6: *t*-SNE plots of sequences found by methods in different rounds (a-c), and of all sequences with a reward greater than the 75th percentile of rewards found by all methods (d). Results are for TfBind8, targetet CRX_R90W_R1. SMW, Evolution, and DbAs-VAE increasingly focus on a single search region, whereas MBO, P3BO and Adaptive-P3BO maintain a diversity of sequences within each batch (Fig. a-c). When showing only sequences that achieved a relatively high reward in Figure 6(d), we see that P3BO and Adaptive-P3BO still find diverse clusters of high-reward sequences. In fact, P3BO and Adaptive-P3BO find sequences in nearly all clusters identified by baseline methods.

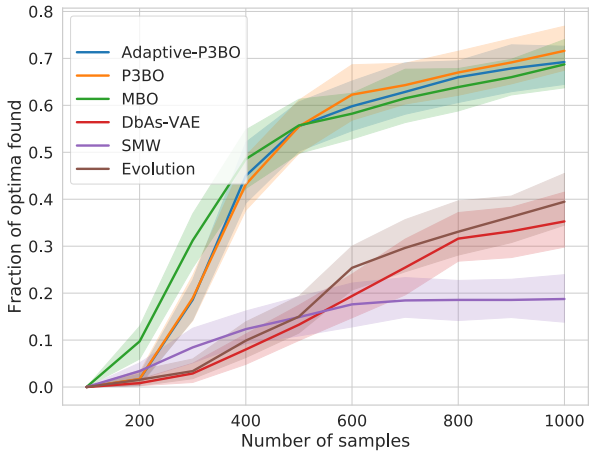


Figure 7: Fraction of optima found by each method averaged over all TfBind8 targets.

Figure 7 reports one of these metrics—the fraction of optima found by each method for TfBind8. P3BO and Adaptive-P3BO find more optima than the best individual method (MBO), and considerably more optima than all remaining methods. Suppl. Figure 1 further shows the mean distance of sequences in proposed batches, and the number of distinct clusters of sequences with a high reward that were found by methods. These results confirm that P3BO and Adaptive-P3BO find diverse sequences with a high reward across optimization problems.

Figure 6 uses *t*-SNE to illustrate the diversity of sequences obtained over time and at the end of optimization for each method. P3BO and Adaptive-P3BO not only find more diverse sequences per batch during the optimization, but also obtain diverse *and* high-reward sequences at the end of the optimization, fulfilling a crucial requirement for real-world sequence validation.

6.5. Ablation Experiments

We perform ablation experiments to better understand the behavior of P3BO and Adaptive-P3BO.

Figure 5 illustrates how the scoring of algorithms affects the number of sequences sampled from each algorithm class and the number of instances per class for one PfamHMM problem. Adaptive-P3BO correctly identifies DbAs-VAE as the best performing algorithm (see Figure 4, PfamHMM), and hence samples more sequences from DbAs-VAE and enriches for the number of DbAs-VAE instances in the population. We observe the same behavior on other optimization problems (Suppl. Figure 4).

Suppl. Figure 5(a) analyzes the performance of P3BO when removing individual algorithm classes from its population. As expected, the performance of P3BO is most sensitive to the best performing algorithm in its population. However, since the performance of algorithms varies across problems, removing individual algorithms results in a performance drop on some problems but not on others. Suppl. Figure 5 also shows that sharing data between optimization methods enables poorly performing algorithms to learn from the sequences found by well performing algorithms and to find high-quality sequences faster, which results in an overall performance gain of P3BO. This is confirmed visually by Suppl. Figure 6, which shows the t-SNE visualization of sequences proposed by individual algorithms over time with and without data sharing.

Suppl. Figure 6 highlights that Adaptive-P3BO recovers faster than P3BO when starting with a poorly initialized population of algorithms, which it can improve online by evolutionary search as described in Section 3.3.

Finally, we investigate the sensitivity of P3BO to the choice of the scoring function (Suppl. Figure 8, 9), the temperature τ of the softmax scoring function (Suppl. Figure 10), the population size (Suppl. Figure 11), and the batch size of the optimization problem (Suppl. Figure 12).

7. Conclusion

We show on extensive novel *in-silico* benchmarks that standard ML approaches to sequence design lack robustness across optimization problems. By ensembling over algorithms, P3BO increases sample-efficiency, robustness, and the ability to discover diverse optima. Online adaptation of the population of algorithms (Adaptive-P3BO) further improves performance. Efficient ways of selecting the initial population and enabling transfer learning across optimization problems remain crucial open questions. Finally, although P3BO is motivated by biological sequence design, P3BO is also applicable to non-biological black-box optimization problems.

References

Angermueller, C., Dohan, D., Belanger, D., Deshpande, R., Murphy, K., and Colwell, L. Model-based reinforcement learning for biological sequence design. In *ICLR*, 2020.

Arnold, F. H. Design by directed evolution. *Accounts of chemical research*, 31(3), 1998.

Barrera, L. A., Vedenko, A., Kurland, J. V., Rogers, J. M., Gisselbrecht, S. S., Rossin, E. J., Woodard, J., Mariani, L., Kock, K. H., Inukai, S., Siggers, T., Shokri, L., Gordân, R., Sahni, N., Cotsapas, C., Hao, T., Yi, S., Kellis, M., Daly, M. J., Vidal, M., Hill, D. E., and Bulyk, M. L. Survey of variation in human transcription factors reveals prevalent DNA binding changes. *Science*, 2016.

Berman, H. M., Bourne, P. E., Westbrook, J., and Zardecki, C. The protein data bank. In *Protein Structure*. CRC Press, 2003.

Bileschi, M. L., Belanger, D., Bryant, D. H., Sanderson, T., Carter, B., Sculley, D., DePristo, M. A., and Colwell, L. J. Using deep learning to annotate the protein universe. *bioRxiv*, 2019.

Biswas, S., Khimulya, G., Alley, E. C., Esvelt, K. M., and Church, G. M. Low-n protein engineering with data-efficient deep learning. *bioRxiv*, 2020.

Brindle, A. Genetic algorithms for function optimization. 1980.

Brookes, D. H. and Listgarten, J. Design by adaptive sampling. *arXiv preprint arXiv:1810.03714*, 2018.

Brookes, D. H., Busia, A., Fannjiang, C., Murphy, K., and Listgarten, J. A view of estimation of distribution algorithms through the lens of expectation-maximization, 2019.

Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., and Qu, R. Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1695–1724, 2013.

De Boer, P.-T., Kroese, D. P., Mannor, S., and Rubinstein, R. Y. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1), 2005.

de Jongh, R. P., van Dijk, A. D., Julsing, M. K., Schaap, P. J., and de Ridder, D. Designing eukaryotic gene expression regulation using machine learning. *Trends in biotechnology*, 2019.

Du, W. and Li, B. Multi-strategy ensemble particle swarm optimization for dynamic optimization. *Information sciences*, 178(15):3096–3109, 2008.

El-Gebali, S., Mistry, J., Bateman, A., Eddy, S. R., Luciani, A., Potter, S. C., Qureshi, M., Richardson, L. J., Salazar, G. A., Smart, A., et al. The Pfam protein families database in 2019. *Nucleic acids research*, 47(D1), 2018.

Fialho, Á., Schoenauer, M., and Sebag, M. Toward comparison-based adaptive operator selection. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pp. 767–774, 2010.

Finn, R. D., Clements, J., and Eddy, S. R. HMMER web server: interactive sequence similarity searching. *Nucleic acids research*, 39(suppl_2), 2011.

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2), 2018.

Gupta, A. and Zou, J. Feedback GAN (FBGAN) for DNA: A novel feedback-loop architecture for optimizing protein functions. *arXiv preprint arXiv:1804.01694*, 2018.

Hashimoto, T. B., Yadlowsky, S., and Duchi, J. C. Derivative free optimization via repeated classification, 2018.

Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W. M., Donahue, J., Razavi, A., Vinyals, O., Green, T., Dunning, I., Simonyan, K., et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

Khadka, S. and Tumer, K. Evolution-guided policy gradient in reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 1188–1200, 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Le, D. D., Shimko, T. C., Aditham, A. K., Keys, A. M., Longwell, S. A., Orenstein, Y., and Fordyce, P. M. Comprehensive, high-resolution binding energy landscapes reveal context dependencies of transcription factor binding. *Proceedings of the National Academy of Sciences*, 115(16):E3702–E3711, 2018.

Lehman, J. and Stanley, K. O. Exploiting open-endedness to solve problems through the search for novelty. In *ALIFE*, pp. 329–336, 2008.

Leyton-Brown, K., Nudelman, E., Andrew, G., McFadden, J., and Shoham, Y. A portfolio approach to algorithm selection. In *IJCAI*, volume 3, pp. 1542–1543, 2003.

Li, K., Fialho, A., Kwong, S., and Zhang, Q. Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 18(1):114–130, 2013.

Littlestone, N., Warmuth, M. K., et al. *The weighted majority algorithm*. University of California, Santa Cruz, Computer Research Laboratory, 1989.

Liu, G., Zeng, H., Mueller, J., Carter, B., Wang, Z., Schilz, J., Horny, G., Birnbaum, M. E., Ewert, S., and Gifford, D. K. Antibody complementarity determining region design using high-capacity machine learning. *bioRxiv*, 2019.

Maturana, J., Fialho, Á., Saubion, F., Schoenauer, M., and Sebag, M. Extreme compass and dynamic multi-armed bandits for adaptive operator selection. In *2009 IEEE Congress on Evolutionary Computation*, pp. 365–372. IEEE, 2009.

Miller, B. L., Goldberg, D. E., et al. Genetic algorithms, tournament selection, and the effects of noise. *Complex systems*, 9(3):193–212, 1995.

Mockus, J., Tiesis, V., and Zilinskas, A. The application of Bayesian methods for seeking the extremum. *Towards Global Optimization*, 2:117–129, 09 2014.

Neil, D., Segler, M., Guasch, L., Ahmed, M., Plumbley, D., Sellwood, M., and Brown, N. Exploring deep recurrent models with reinforcement learning for molecule design. In *International Conference on Learning Representations Workshop*, 2018.

Pourchot, A. and Sigaud, O. Cem-rl: Combining evolutionary and gradient-based methods for policy search. *arXiv preprint arXiv:1810.01222*, 2018.

Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 2019.

Remmert, M., Biegert, A., Hauser, A., and Söding, J. HHblits: lightning-fast iterative protein sequence searching by HMM-HMM alignment. *Nature methods*, 9(2), 2012.

Sample, P. J., Wang, B., Reid, D. W., Presnyak, V., McFadyen, I. J., Morris, D. R., and Seelig, G. Human 5 UTR design and variant effect prediction from a massively parallel translation assay. *Nature biotechnology*, 37(7), 2019.

Tang, K., Peng, F., Chen, G., and Yao, X. Population-based Algorithm Portfolios with automated constituent algorithms selection. *Information Sciences*, 279:94–104, 2014.

unter Rudolph, G. Global optimization by means of distributed evolution strategies. *Schwefel and M anner*, 2035, 1958.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575 (7782):350–354, 2019.

Wang, Y., Wang, H., Liu, L., and Wang, X. Synthetic promoter design in escherichia coli based on generative adversarial network. *BioRxiv*, 2019.

Wu, G., Mallipeddi, R., and Suganthan, P. N. Ensemble strategies for population-based optimization algorithms–a survey. *Swarm and evolutionary computation*, 44, 2019a.

Wu, Z., Kan, S. J., Lewis, R. D., Wittmann, B. J., and Arnold, F. H. Machine learning-assisted directed protein evolution with combinatorial libraries. *Proceedings of the National Academy of Sciences*, 116(18), 2019b.

Yang, K. K., Wu, Z., and Arnold, F. H. Machine-learning-guided directed evolution for protein engineering. *Nature methods*, 2019.